nag_mv_ordinal_multidimscale (g03fcc)

1. Purpose

nag_mv_ordinal_multidimscale (g03fcc) performs non-metric (ordinal) multidimensional scaling.

2. Specification

```
#include <nag.h>
#include <nagg03.h>

void nag_mv_ordinal_multidimscale(Nag_ScaleCriterion type, Integer n,
         Integer ndim, double d[], double x[], Integer tdx,
         double *stress, double dfit[], Nag_E04_Opt *options,
         NagError *fail)
```

3. Description

For a set of n objects, a distance or dissimilarity matrix D can be calculated such that d_{ij} is a measure of how 'far apart' objects i and j are. If p variables x_k have been recorded for each observation this measure may be based on Euclidean distance, $d_{ij} = \sum_{k=1}^{p}(x_{ki} - x_{kj})^2$, or some other calculation such as the number of variables for which $x_{kj} \neq x_{ki}$. Alternatively, the distances may be the result of a subjective assessment. For a given distance matrix, multidimensional scaling produces a configuration of n points in a chosen number of dimensions, m, such that the distance between the points in some way best matches the distance matrix. For some distance measures, such as Euclidean distance, the size of distance is meaningful, for other measures of distance all that can be said is that one distance is greater or smaller than another. For the former, metric scaling can be used, see nag_mv_prin_coord_analysis (g03fac), for the latter, a non-metric scaling is more appropriate.

For non-metric multidimensional scaling, the criterion used to measure the closeness of the fitted distance matrix to the observed distance matrix is known as $STRESS$. $STRESS$ is given by,

$$\sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{i-1}(\hat{d}_{ij} - \tilde{d}_{ij})^2}{\sum_{i=1}^{n}\sum_{j=1}^{i-1}\hat{d}_{ij}^2}}$$

where $\hat{d}_{ij}^2$ is the Euclidean squared distance between points i and j and $\tilde{d}_{ij}$ is the fitted distance obtained when $\hat{d}_{ij}$ is monotonically regressed on d_{ij}, that is, $\tilde{d}_{ij}$ is monotonic relative to d_{ij} and is obtained from $\hat{d}_{ij}$ with the smallest number of changes. So $STRESS$ is a measure of by how much the set of points preserve the order of the distances in the original distance matrix. Non-metric multidimensional scaling seeks to find the set of points that minimize the $STRESS$.

An alternate measure is squared $STRESS$, $SSTRESS$,

$$\sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{i-1}(\hat{d}_{ij}^2 - \tilde{d}_{ij}^2)^2}{\sum_{i=1}^{n}\sum_{j=1}^{i-1}\hat{d}_{ij}^4}}$$

in which the distances in $STRESS$ are replaced by squared distances.

In order to perform a non-metric scaling, an initial configuration of points is required. This can be obtained from principal co-ordinate analysis, see nag_mv_prin_coord_analysis (g03fac). Given an initial configuration, nag_mv_ordinal_multidimscale uses the optimization routine nag_opt_conj_grad (e04dgc) to find the configuration of points that minimizes $STRESS$ or $SSTRESS$. The routine nag_opt_conj_grad (e04dgc) uses a conjugate gradient algorithm. nag_mv_ordinal_multidimscale will find an optimum that may only be a local optimum, to be more sure of finding a global optimum several different initial configurations should be used; these can be obtained by randomly perturbing the original initial configuration using routines from Chapter g05.

4. Parameters

type

Input: indicates whether $STRESS$ or $SSTRESS$ is to be used as the criterion.

If **type** = **Nag_Stress**, $STRESS$ is used.
If **type** = **Nag_SStress**, $SSTRESS$ is used.

Constraint: **type** = **Nag_Stress** or **Nag_SStress**.

n

Input: the number of objects in the distance matrix , n.

Constraint: **n** > **ndim**.

ndim

Input: the number of dimensions used to represent the data, m.

Constraint: **ndim** ≥ 1.

d[n∗(n−1)/2]

Input: the lower triangle of the distance matrix D stored packed by rows. That is **d**$[(i-1)*(i-2)/2+j-1]$ must contain d_{ij} for $i = 2, 3, \ldots, n$; $j = 1, 2, \ldots, i-1$. If d_{ij} is missing then set $d_{ij} < 0$; For further comments on missing values see Section 6.

x[n][tdx]

Input: the ith row must contain an initial estimate of the co-ordinates for the ith point, $i = 1, 2, \ldots, n$. One method of computing these is to use nag_mv_prin_coord_analysis (g03fac).

Output: the ith row contains m co-ordinates for the ith point, $i = 1, 2, \ldots, n$.

tdx

Input: the last dimension of the array **x** as declared in the calling program.
Constraint: **tdx** $\geq$ **ndim**.

stress

Output: the value of $STRESS$ or $SSTRESS$ at the final iteration.

dfit[2∗n∗(n−1)]

Output: auxiliary outputs. If **type** = **Nag_Stress**, the first $n(n-1)/2$ elements contain the distances, $\hat{d}_{ij}$, for the points returned in **x**, the second set of $n(n-1)/2$ contains the distances $\hat{d}_{ij}$ ordered by the input distances, d_{ij}, the third set of $n(n-1)/2$ elements contains the monotonic distances, $\tilde{d}_{ij}$, ordered by the input distances, d_{ij} and the final set of $n(n-1)/2$ elements contains fitted monotonic distances, $\tilde{d}_{ij}$, for the points in **x**. The $\tilde{d}_{ij}$ corresponding to distances which are input as missing are set to zero. If **type** = **Nag_SStress**, the results are as above except that the squared distances are returned.

Each distance matrix is stored in lower triangular packed form in the same way as the input matrix D.

options

Input/Output: a pointer to a structure of type Nag_E04_Opt whose members are optional parameters for nag_opt_conj_grad (e04dgc). These structure members offer the means of adjusting some of the parameter values of the algorithm and on output will supply further details of the results. You are referred to the nag_opt_conj_grad (e04dgc) document for further details.

The default values used by nag_mv_ordinal_multidimscale when the options parameter is set to the NAG defined null pointer, `E04_DEFAULT`, are as follows:

options.optim_tol = 0.00001;
options.print_level = **Nag_NoPrint**;
options.list = **FALSE**;
options.verify_grad = **FALSE**;
options.max_iter = MAX(50, **n**∗**ndim**).

If a different value is required for any of these four structure members or if other options available in nag_opt_conj_grad (e04dgc) are to be used, then the structure **options** should be declared and initialised by a call to nag_opt_init (e04xxc) and supplied as an argument to nag_mv_ordinal_multidimscale. In this case, the structure members listed above except for **list** will have the default values as specified above; **options.list** = **TRUE** in this case.

fail
The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_BAD_PARAM
On entry, parameter **type** had an illegal value.

NE_INT_ARG_LT
On entry, **ndim** must not be less than 1: **ndim** = ⟨*value*⟩.

NE_2_INT_ARG_LE
On entry, **n** = ⟨*value*⟩ while **ndim** = ⟨*value*⟩.
These parameters must satisfy **n** > **ndim**.

NE_2_INT_ARG_LT
On entry, **tdx** = ⟨*value*⟩ while **ndim** = ⟨*value*⟩.
These parameters must satisfy **tdx** ≥ **ndim**.

NE_NEG_OR_ZERO_ARRAY
All elements of array **d** ≤ 0.0.
Constraint: At least one element of **d** must be positive.

NE_ALLOC_FAIL
Memory allocation failed.

NE_INTERNAL_ERROR
An internal error has occurred in this function.
Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

Additional error messages are output if the optimization fails to converge or if the options are set incorrectly, Details of these can be found in the nag_opt_conj_grad (e04dgc) document.

6. Further Comments

Missing values in the input distance matrix can be specified by a negative value and providing there are not more than about two thirds of the values missing, the algorithm may still work. However, the routine nag_mv_prin_coord_analysis (g03fac) does not allow for missing values so an alternative method of obtaining an initial set of co-ordinates is required. It may be possible to estimate the missing values with some form of average and then use nag_mv_prin_coord_analysis (g03fac) to give an initial set of co-ordinates.

6.1. Accuracy

After a successful optimization, the relative accuracy of $STRESS$ should be approximately ϵ, as specified by **options.optim_tol**.

6.2. References

Chatfield C and Collins A J (1980) *Introduction to Multivariate Analysis* Chapman and Hall.
Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press.

7. See Also

nag_mv_prin_coord_analysis (g03fac)
nag_opt_conj_grad (e04dgc)

8. Example

The data, given by Krzanowski (1990), are dissimilarities between water vole populations in Europe. Initial estimates are provided by the first two principal co-ordinates computed by nag_mv_prin_coord_analysis (g03fac). The two dimension solution is computed using nag_mv_ordinal_multidimscale.

8.1. Program Text

```
/* nag_mv_ordinal_multidimscale (g03fcc) Example Program.
 *
 * Copyright 1998 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 *
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg01.h>
#include <nagg03.h>

#define NMAX 14
#define MMAX 2
#define NNMAX NMAX*(NMAX-1)/2

#define X(I,J) x[(I-1)*NMAX + (J-1)]
#define XTMP(I) xtmp[(I)-1]
#define YTMP(I) ytmp[(I)-1]



main()
{
  double d[NNMAX], dfit[4*NNMAX], wk[NNMAX+15*NMAX*MMAX],
  x[NMAX*NMAX];
  double stress;

  Integer ndim;
  Integer i, j, n;
  Integer nn;
  Integer tdx = NMAX;

  char char_type[2];
  Nag_ScaleCriterion type;

  Vprintf("g03fcc Example Program Results\n\n");

  /*  Skip heading in data file */
  Vscanf("%*[^\n]");

  Vscanf("%ld",&n);
  Vscanf("%ld",&ndim);
  Vscanf("%s",char_type);

  if (n <= NMAX)
    {
      nn = n * (n - 1) / 2;
      for (i = 1; i <= nn; ++i)
        Vscanf("%lf",&d[i-1]);

      g03fac(Nag_LargeEigVals, n, d, ndim, x, tdx, wk, NAGERR_DEFAULT);
      if (*char_type == 'T')
        type = Nag_Stress;
      else
        type = Nag_SStress;

      g03fcc(type, n, ndim, d, x, tdx, &stress, dfit,
             E04_DEFAULT, NAGERR_DEFAULT);

      Vprintf("\n          STRESS = %13.4e\n\n",stress);
      Vprintf("Co-ordinates\n\n");

      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= ndim; ++j)
            Vprintf("%10.4f",X(i,j));
```

```
          Vprintf("\n");
        }
      exit(EXIT_SUCCESS);
    }
  else
    {
      Vprintf("Incorrect input value of n.\n");
      exit(EXIT_FAILURE);
    }
}
```

8.2. Program Data

```
g03fcc Example Program Data

14 2 T

0.099
0.033 0.022
0.183 0.114 0.042
0.148 0.224 0.059 0.068
0.198 0.039 0.053 0.085 0.051
0.462 0.266 0.322 0.435 0.268 0.025
0.628 0.442 0.444 0.406 0.240 0.129 0.014
0.113 0.070 0.046 0.047 0.034 0.002 0.106 0.129
0.173 0.119 0.162 0.331 0.177 0.039 0.089 0.237 0.071
0.434 0.419 0.339 0.505 0.469 0.390 0.315 0.349 0.151 0.430
0.762 0.633 0.781 0.700 0.758 0.625 0.469 0.618 0.440 0.538 0.607
0.530 0.389 0.482 0.579 0.597 0.498 0.374 0.562 0.247 0.383 0.387 0.084
0.586 0.435 0.550 0.530 0.552 0.509 0.369 0.471 0.234 0.346 0.456 0.090 0.038
```

8.3. Program Results

```
g03fcc Example Program Results


          STRESS =    1.2557e-01

Co-ordinates

    0.2060    0.2439
    0.1063    0.1418
    0.2224    0.0817
    0.3032    0.0355
    0.2645   -0.0698
    0.1554   -0.0435
   -0.0070   -0.1612
    0.0749   -0.3275
    0.0488    0.0289
    0.0124   -0.0267
   -0.1649   -0.2500
   -0.5073    0.1267
   -0.3093    0.1590
   -0.3498    0.0700
```